Tero: Offloading CDN Traffic to Massively Distributed Devices

ICDCN 2024

Juan Vanerio - University of Vienna
Lily Hügerich - TU Berlin
Stefan Schmid - TU Berlin & University of Vienna

January 5, 2024

universität wien

Technische Universität Berlin

Content Delivery Networks (CDN)

- A network of geographically-distributed cache servers delivering content to users.
- Crucial component in delivering content quickly to users around the world.
  - Video: 65% of all Internet traffic, mostly served from CDNs.
- **Content Distribution**: replicate and cache content (images, videos, files) close to users.
- **Reduced Latency**: Smaller RTTs result in faster loading times.
- **Reliability**: Content redundancy via replication enhances its reliability and availability.

How a CDN works

- **Origin Servers**:
  - Central repository of original content.
  - Hosts dynamic content and authoritative copies.
- **Edge Servers**:
  - Deliver frequently requested (*popular*) content.
  - Located at the network's periphery, close to end-users.
  - Handle load locally.
- User requests are routed to **nearest servers** for swift access.
- **What if the EDGE server is overwhelmed? What happens during demand bursts?**
  - **Idea:** Balance the load using further distribution!

Three-Tier CDN Architecture

Introduce a third tier in CDNs: **Regional *Caching Devices***

- **Highly distributed**: Placed in block cabinets, base-stations, Set-top boxes, etc.
- Typically limited in storage, bandwidth, processing power.
- Serve more traffic and alleviate EDGE server's load.
  - Demand bursts and spikes!
- Local EDGE server **redirects content requests** to specific devices. **How?**

Requirements for a three-tier CDN system

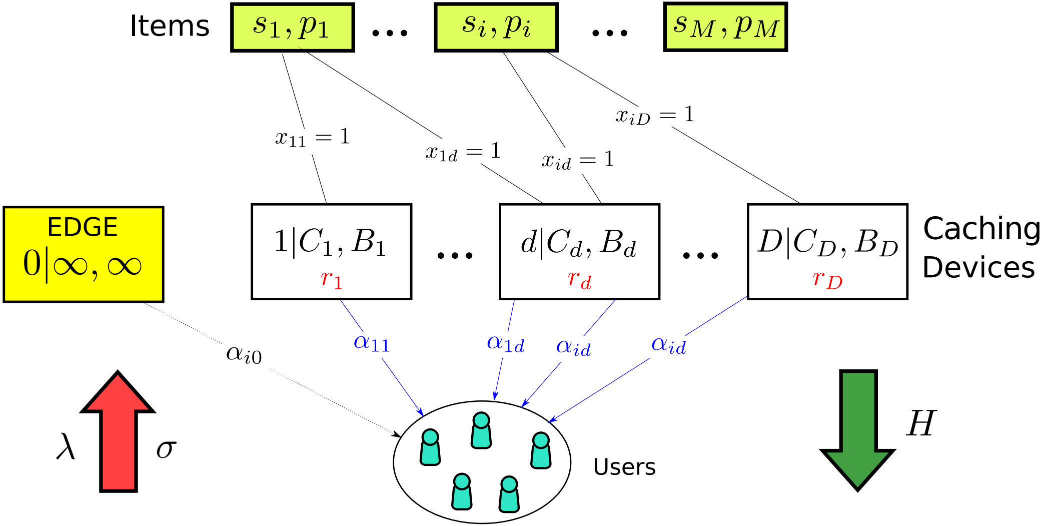
- Request routing must be performed online
- Two options for content placement in caching devices:
  - On-demand (online)
  - Prefetching (offline, periodical)
- prefetching being preactive has potential for better performance...
  - ... if the popular content can be accurately predicted!
  - trade-off between accuracy and resource-usage.
- Existing approaches predict long term popularities inferred from large amounts of data using complex, slow, resource intensive algorithms.

Can we make it simpler?

Design goals for a three-tier CDN system

- Decrease EDGE server **traffic** with respect to realizable baselines.
- **Fast** decisions (order of minutes or less).
- Leverage item **popularity prediction**.
- Provide **service quality** to requests.

Notation



Items
s_1, p_1
...
s_i, p_i
...
s_M, p_M
$x_{11} = 1$
$x_{1d} = 1$
$x_{id} = 1$
$x_{iD} = 1$
EDGE
$0|\infty, \infty$
$1|C_1, B_1$
r_1
...
$d|C_d, B_d$
r_d
...
$D|C_D, B_D$
r_D
Caching Devices
α_{i0}
α_{11}
α_{1d}
α_{id}
α_{id}
λ
σ
Users
H

Notation Summary

- (Content) item i: Size s_i, popularity p_i.
- Caching device d: Storage capacity C_d, bandwidth B_d, concurrent requests: r_d.
  - $d = 0$ represents the EDGE server.
- H throughput from the cache devices.
- $x_{id} \in \{0, 1\}$: Allocation control variable; $x_{id} = 1$ if item i is to be placed on device d.
- $\sigma = (\sigma_1, \dots, \sigma_N)$: Online request sequence up to time T.
- $\lambda = \frac{N}{T}$ request process intensity
- $\alpha_{id} \in [0, 1]$: demand fraction to item i served from d.
- π routing policy: $\pi(\mathbf{x}, \sigma) \Rightarrow \alpha$
- δ: Minimum acceptable bandwidth per request (service quality).

Main Model

- $\min\limits_{X,A} \quad H_s = \sum_{i=1}^{M} s_i N_{i0}$ (1)
- Subject to:
  - **Demand conservation**: $\sum_{d=0}^{D} \alpha_{id} = 1 \quad \forall i \in [1, M]$ (2)
  - **Storage capacity**: $\sum_{i=1}^{M} x_{id} s_i \leq C_d \quad \forall d \in [1, D]$ (3)
  - **Bandwidth**: $\sum_{i=1}^{M} s_i N_{id} \leq TB_d \quad \forall d \in [1, D]$ (4)
  - **Admission**: $r_d \leq R_d = \frac{B_d}{\delta} \quad \forall d \in [1, D]$ (5)
  - $0 \leq \alpha_{id} \leq x_{id} \leq 1$ (6), $x_{id} \in \{0, 1\}, \alpha_{id} \in [0, 1]$ (7), $\forall i \in [1, M], d \in [1, D]$

Solving directly resulted in **long execution times**.

How to approach the problem?

- **Idea 1**: Split into sub-problems.
- **Idea 2**: Identify properties of optimum and design heuristics.
- **Observation**: Total traffic demand can be split into:
  - Traffic H served from the devices,
  - Traffic served from EDGE because uncached,
  - **Excess** traffic served from EDGE server because resource exhaustion in caches.
    - Happens when all devices hosting replicas of an item are already serving at full capacity.
  - For a given allocation $\mathbf{x}$, the optimal routing is the one that *minimizes excess traffic*.
- **Heuristic**: Place requests on devices with lower number of concurrent requests.
- **Problem**: current usage r_d is not directly observable!

Allocation Model

- $\max_{\mathbf{x}} \quad H(\mathbf{x}|\pi) = \sum_{i=1}^{M} \lambda s_i p_i (1 - \alpha_{i0}|\pi)$
- Subject to:
  - **Storage capacity**: $\sum_{i=1}^{M} x_{id} s_i \leq C_d \, \forall d \in [D]$
  - **Bandwidth**: $\lambda \sum_{i=1}^{M} s_i p_i \alpha_{id} \leq B_d \, \forall d \in [D]$
  - **Demand conservation**: $\sum_{d=0}^{D} \alpha_{id} = 1 \quad \forall i \in [1, M]$
  - **Routing policy** π
  - $0 \leq \alpha_{id} \leq x_{id} \leq 1, \forall i \in [M], d \in [D]$
- Note: Still a hard problem, but simpler than the full one.

Optimality Necessary Conditions

- No item replacement or swapping improves performance.
- Replicas observe fractional traffic, number of replicas is minimal.
- The following condition holds:

$$\left(B_d - \lambda \sum_{i=1}^{M} s_i p_i \alpha_{id}\right) \alpha_{id} \left(\alpha_{id} - x_{id}\right) = 0 \tag{8}$$

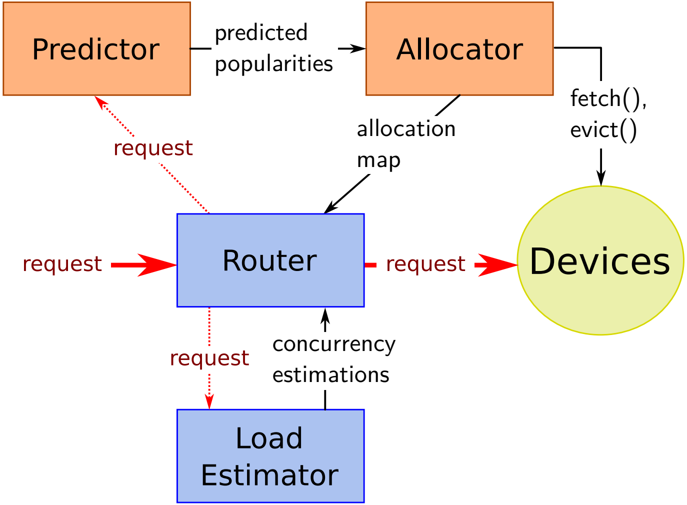
- All devices hosting a replica operate at full bandwidth.
  - Oversubscription must be minimum (avoids excess)
- If a device has remaining bandwidth, it hosts no replicas.
- **Take away**: devices should fill up its storage and bandwidth capacities.

Performance Upper Bound

Single Device Model (SDM)

- Consider a single caching device with:
  - $C = \sum_D C_d$: aggregated storage capacity,
  - $BW = \sum_D B_d$: aggregated bandwidth capacity.
- Caching by popularity: (Usually) approaches optimal performance well.

Full System Overview – as per Simulator



(Popularity and Intensity) Predictor

- Learning source: historical requests log.
  - May be short!
  - Counting process: requests into bins (time slots).
  - **Hypothesis**: Band-limited signal.
- Predictions should have small footprint (costly EDGE server resources)
  - **Pre-caching**: Ignore items requested just once.
  - Band-limited signals experience little change on short time intervals.
- Approach: **per-item moving average** over the last time slots.
- **BASELINE**: True popularity.
- **Question**: Can it perform better than more sophisticated methods? See paper.

Allocator

Offline

- Decides a static allocation for next time slot.
- Uses on requests predictions as input.

Online

- Quick, dynamic adjustments on anomalies and demand bursts.

BASELINES:

- popularity based.
- popularity-proportional replica assignment.

Allocator – Offline

Algorithm 1 Offline Allocation - Deterministic and run periodically.

Initialize $v_i \leftarrow \lambda p_i s_i \quad \forall i,\ b_d \leftarrow B_d \quad \forall d,\ c_d \leftarrow C_d \quad \forall d \in [D]$
while possible **do**
    $j \leftarrow \arg\max_i \{v\}$
    Let D' be the set of devices s.t. $c_d \geq s_j$ and $j \notin d' \forall d' \in D'$.
    Allocate j in $d^* = \arg\max_{d' \in D'} \dfrac{b_{d'}}{c_{d'}}$.
    $c_{d^*} \leftarrow c_{d^*} - s_j$ ▷ Update remaining space.
    if $b_{d^*} \leq v_j$ **then** ▷ Item incidence don't fit in device
        $v_j \leftarrow v_j - b_{d^*},\ b_d \leftarrow 0$ ▷ Update item incidence, deplete device
    else
        $v_j \leftarrow 0,\ b_{d^*} \leftarrow b_{d^*} - v_j$ ▷ Update remaining bandwidth, deplete item
    end if
end while

Concurrency Estimation Module

- Blind (no feedback) – avoids monitoring overhead.
- Estimates proxy of ongoing requests on each device.
  - On each incoming request:
    - Gets tracked request size $\hat{s}$ per candidate device.
    - Estimates **makespan** time as makespan $\leftarrow$ makespan $+ \hat{s}/B_d$
    - Computes threshold to warn of possible service violation: threshold $= \frac{\hat{s}}{2\delta}$
  - The threshold comes from assuming random arrivals at unknown times.
- Devices marked above the makespan threshold are ignored from routing.
- **BASELINE**: True concurrency information from the devices.

Router

- Minimum estimated makespan first (Deterministic).
  - Allocation aware.
- Routes to the EDGE server:
  - If Requested item is not cached anywhere (Unallocated), or
  - Load Estimator warns all item's replica hosting devices are overloaded.
  - Finally update the 'Counting'-LRU cache for detecting anomalies.
- **BASELINE**: Allocation-aware random router.

Evaluation

- Implemented an event-driven simulator based on SimPy for request processing.
- Performance evaluation on a real-world traffic trace and a synthetic one:
  - On each, assessment of prediction performance.
  - On each, ablation study for each system component.

Sources of Traffic Traces

Log-based:

- Injection of requests from logs files.

Synthetic:

- Poisson source with controllable intensity
- Potential (Zipfian-like) popularity distribution
- Same size distribution as log-based.

Devices

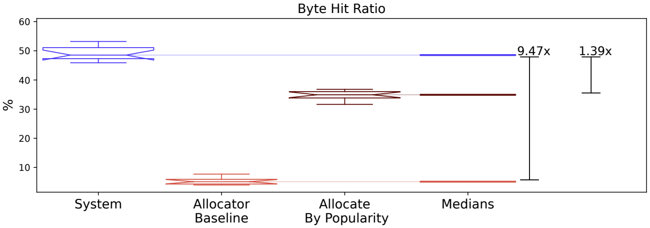
- Size and bandwidth limited cache (except EDGE).
- Processor-sharing discipline.
- Devices download content from EDGE server.
  - Costly and not instantaneous.
- Updates router allocation map only on call return (granular).

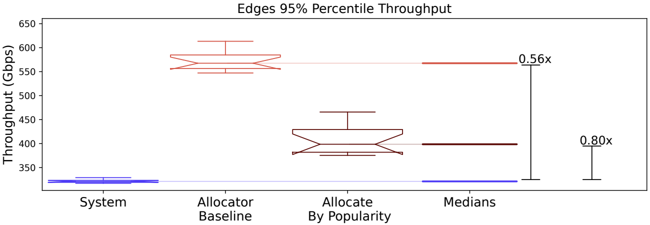
Setup for Synthetic (Poisson) Experiments

No. EDGE Servers	1
Catalog Size	100000
Request Intensity	10000 req/s
QoS Threshold (δ)	1 Mbps
Simulated Time	600 s
Warmup Time	120 s
Seeds	12
Prediction Interval	15 s
Prediction Method	Moving average of last 300s
Item Sizes	100 KB - 1 GB (avg: 7 MB)

Fleet	No. Devices	BW	Download BW	Storage
1	7000	50 Mbps	50Mbps	32 GB
2	14000	20Mbps	20 Mbps	32 GB

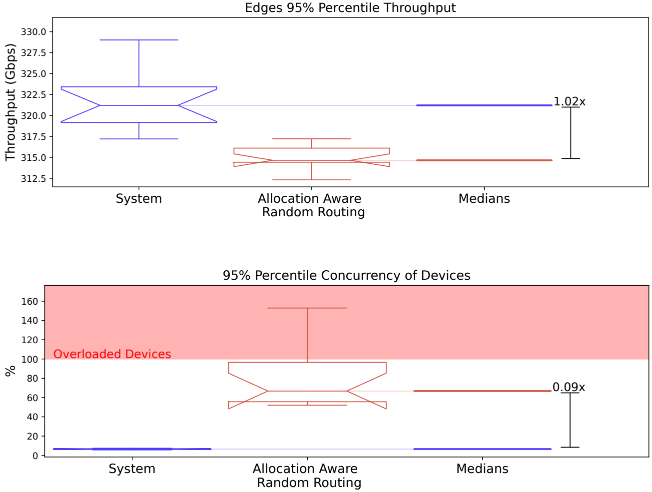
Synthetic Source - Allocator





- System capacity smaller than demand.
- 44% less traffic than baseline.
- 20% less traffic than popularity method.
- Almost 10x larger BHR than baseline.

Synthetic Source - Router

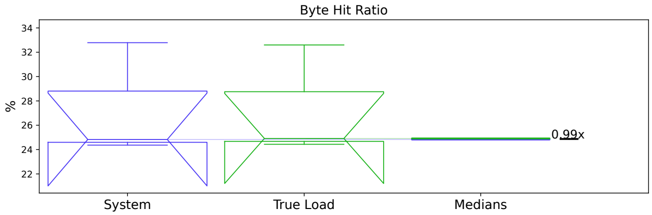


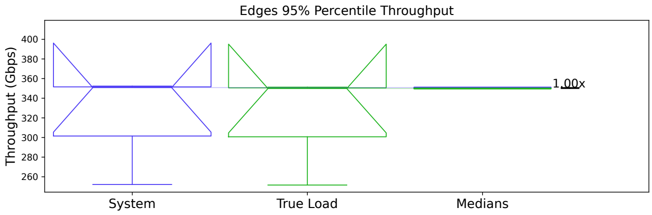
- Similar traffic than baseline.
- 9x reduction in concurrency.
- Strong reduction of temporal QoS violations.

Real-World Log-Based Source Evaluations

- Identical system configuration than in Poisson experiments.
- Intensity, popularity distribution and catalog size are implicit and dynamic
- 30 minutes (19:30-20), 15 minutes (16:00-16:15), and 10 minutes (8:00-8:10) of log traffic.
- No randomization seeds.

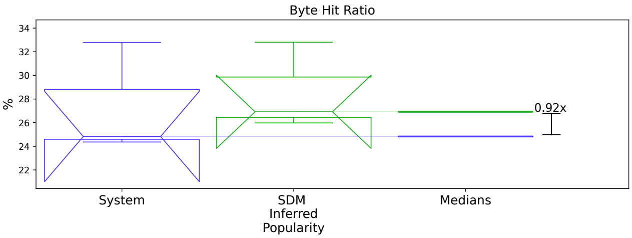
Real-World Source - Load Estimator

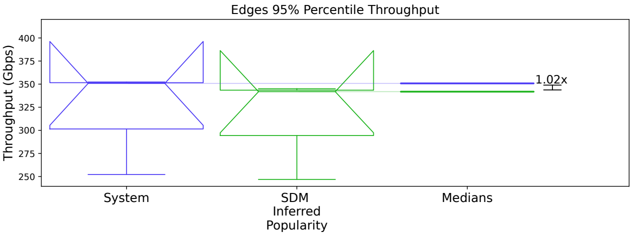




- Less than 1% difference in traffic against true concurrency.
- Less than 1% difference in BHR against true concurrency.

Real-World Source - Single Device Model





- Comparable performance to SDM.

Results Summary – Real-World Source

- Execution times for allocation computations under 10s in the experiments.
- Allocator: 23% traffic reduction wrt baseline system and similar performance to popularity-based allocation (undersaturated setting).
- Router: Comparable results to baseline with 48% less in 95% percentile concurrency.
- Load Estimator: Less than 1% difference in traffic against knowing the true concurrency.
- Comparable performance to SDM using inferred popularity.

Conclusions

Contributions:

- **TERO** is a fast, lightweight and centralized control system for the three-tier CDN.
- A formal model of the three-tier CDN and characteristics of optimal solutions.
- Novel allocation mechanism leveraging short-term item popularity prediction.
- Provides service quality for each request.
- Achievements:
  - Decreases EDGE server traffic with respect to realizable baselines.
  - **Fast**: Computes allocations in seconds for thousands of devices.
  - Good performance wrt baselines and upper bounds on different workloads.
- Router and load estimations are online and parallelizable.
- There is room for further improvements (Future work: Machine Learning enhancements).

Backup slides

Notation

- $i \in [M]$ is a content object or item.
  - Size s_i
  - Popularity p_i
- $d \in [D]$ is a 302 cache device
  - Storage capacity C_d
  - Upload bandwidth B_d
  - Number of concurrent requests r_d
  - Download bandwidth.
  - $d = 0$ represents the EDGE server.
- H throughput from the cache devices.
- $x_{id} \in \{0, 1\}, \forall i \in [M], d \in [D]$: Allocation control variable; $x_{id} = 1$ if item i is to be placed on device d.

Notation

- $\sigma = (\sigma_1, \ldots, \sigma_N)$: Request sequence up to time T. Each request $\sigma_t \in [N]$. The sequence σ is revealed in an online manner.
- $\lambda = \frac{N}{T}$ request process intensity
- $\alpha_{id} \in [0,1], \forall i \in [M], d \in [D]$: fraction of traffic demand to item i served from d.
  - $\sum_{d=1}^{D} \alpha_{id} = 1 - \alpha_{i0} \quad \forall i \in [M]$
- π routing policy: $\pi(\mathbf{x}, \sigma) \Rightarrow \alpha$
- δ: Minimum acceptable bandwidth per request.

(Popularity and Intensity) Predictor

- Learning source: historical requests log.
  - Counting process: requests into bins (time slots).
  - **Hypothesis**: Band-limited signal.
- Algorithms to predict requests per item on next snapshot:
  - **Simple_x**: per item incidence moving average on the last x 1ms time slots.
  - **Linear**: based on the last x time slots.
  - Others: BHT-ARIMA, LFO, ARMA.
- Uses **pre-caching**: Ignore items requested just once.
- **BASELINE: True popularity**
- Results already presented in previous presentations.

Characterization

- Variant of **Capacitated Facility Location Problem** (NP-Hard).
  - **Basic ver.**: Choose facility locations to minimize transportation costs under demand.
  - Items map to customers, devices to facilities, costs to sizes.
  - Doubly-Capacitated: bandwidth and storage.
  - Plus an admission constraint (QoS).
  - Fully overlapped coverage.
- Attempts at solving directly resulted in excessively **long execution times**.

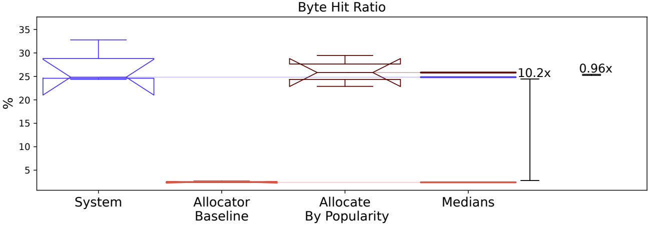
Allocator – Offline

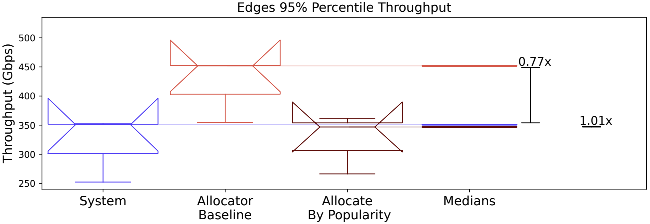
- Brownfield deployment:
  - Sticky allocation
  - Candidate selection offsets $\Rightarrow$ increased device diversity.
  - Device selection variant:
    Attempt to allocate at $\arg\max_{D'} \frac{b_{d*}}{c_{d*}}$ s.t. $p_j > \frac{b_{d*}}{c_{d*}}$, otherwise in $\arg\max_{D'} \frac{b_{d*}}{c_{d*}}$

Online Allocation – Online

- Detects anomalies on expected request prediction.
- Anomaly: Too many requests being directed to the EDGE server.
- (Custom) 'Counting'-LRU algorithm in the Router:
  - Tracks z_i number of requests sent to EDGE, per item.
  - If z_i exceeds a threshold, triggers allocation of i into device $d' \in \{d \in D : i \notin d\}$ with the lowest *makespan* estimation.
  - d' evicts according to LRU-k policy.

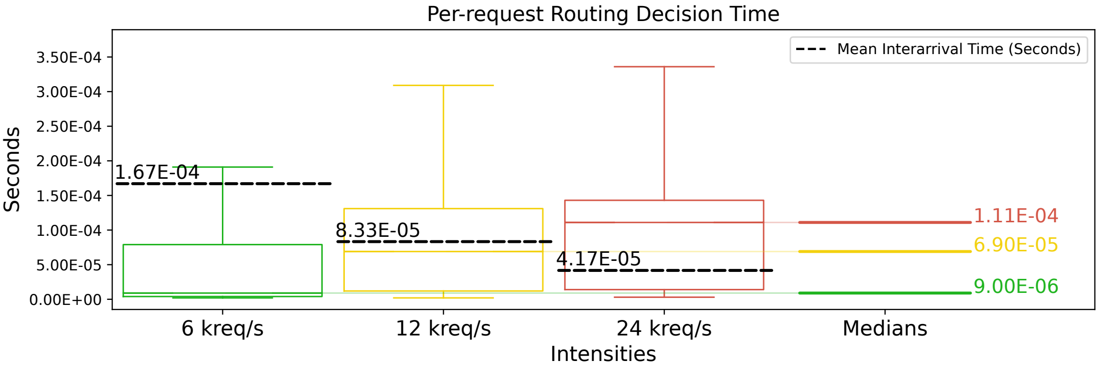
Real-World Source - Allocator





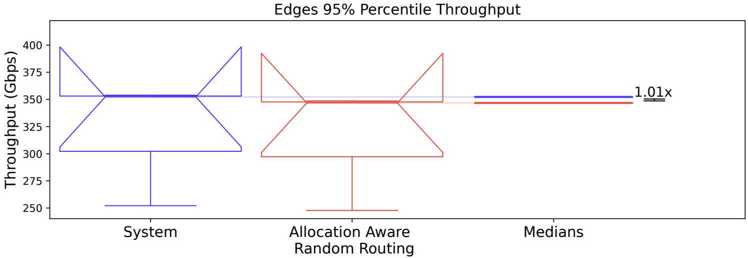
- Undersaturated System: capacity exceeds cacheable traffic demand.
- 23% traffic reduction wrt baseline system
- Similar performance to popularity-based allocation (due to undersaturation)

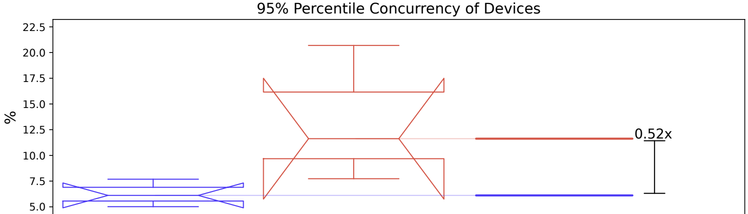
Synthetic Source - Router Times



- Undersaturated (left) and critical (middle) systems route in real time.
- Single-threaded implementation.
- Parallelization is feasible and will lead to shorter times.

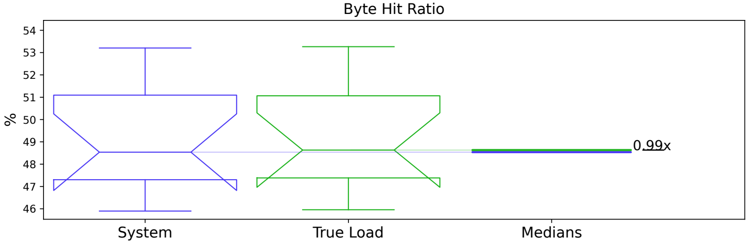
Real-World Source - Router

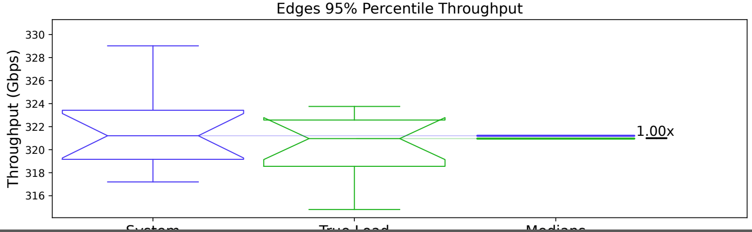




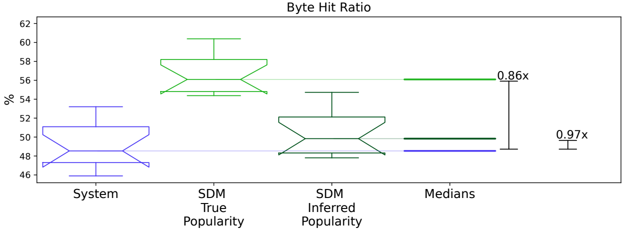
- Comparable results to baseline with 48% less concurrency.

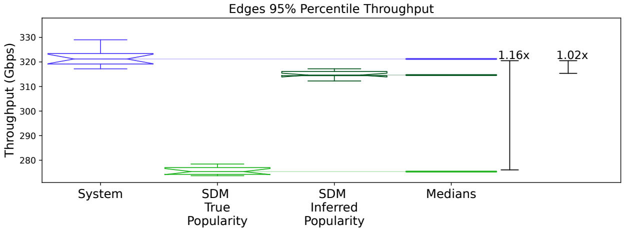
Poisson Source Evaluations - Load Estimator





- Less than 1% difference in traffic against true concurrency.
- Less than 1% difference in BHR against true concurrency.





- Comparable performance to SDM using inferred popularity.
- Performance gap: 16% more traffic and 14% less BHR than ideal system.
- Room for improvement.